Automatic obstacle avoidance of quadrotor UAV via CNN-based learning

Article (Accepted Version)

Automatic Obstacle Avoidance of Quadrotor UAV via CNN-based Learning

Xi Dai[a], Yuxin Mao[a], Tianpeng Huang[a], Na Qin[a], Deqing Huang[a,*], Yanan Li[b]

[a]*School of Electrical Engineering, Southwest Jiaotong University, Chengdu, China*
[b]*School of Engineering and Informatics, University of Sussex, Brighton, BN1 9RH, UK.*

Abstract

In this paper, a CNN-based learning scheme is proposed to enable a quadrotor unmanned aerial vehicle (UAV) to avoid obstacles automatically in unknown and unstructured environments. In order to reduce the decision delay and to improve the robustness for the UAV, a two-stage end-to-end obstacle avoidance architecture is designed, where a forward-facing monocular camera is used only. In the first stage, a convolutional neural network (CNN)-based model is adopted as the prediction mechanism. Utilizing three effective operations, namely depthwise convolution, group convolution and channel split, the model predicts the steering angle and the collision probability simultaneously. In the second stage, the control mechanism maps the steering angle to an instruction that changes the yaw angle of the UAV. Consequently, when the UAV encounters an obstacle, it can avoid collision by steering automatically. Meanwhile, the collision probability is mapped as a forward speed to maintain the flight or stop going forward. The presented automatic obstacle avoidance scheme of quadrotor UAV is verified by several indoor/outdoor tests, where the feasibility and efficacy have been demonstrated clearly. The novelties of the method lie in its low sensor requirement, light-weight network structure, strong learning ability and environmental adaptability.

Keywords: Obstacle avoidance, Unmanned aerial vehicle, Convolutional neural network, Collision probability.

1. Introduction

Automatic obstacle avoidance is crucial to many autonomous systems, such as unmanned aerial vehicles (UAVs). Developing intelligent and autonomous quadrotors is on the centre stage of UAV research in recent years, due to their usefulness in aiding safety and intuitive control when working in different scenarios, e.g. surveillance, mapping, construction monitoring, delivery, and traffic monitoring. In order to adapt to working environments and complete the above tasks, perception, control, and localization should be addressed in a UAV control system simultaneously. To ensure the best performance in a social environment

*Corresponding author
Email address: elehd@swjtu.edu.cn (Deqing Huang)

which is usually unstructured and highly dynamic, a UAV is required to interact safely with other agents that are presented in the environment, e.g. cars and pedestrians (see Figure 1). To address the obstacle avoidance problem,

Figure 1: **A UAV flies through a grove of trees.** The work focuses on obstacle avoidance of a UAV in a social environment.

a quadrotor is required to sense the surrounding environments and react to them quickly. Several widely recognized approaches are simultaneous location and mapping (SLAM) [1], structure from motion (SfM) [2] and learning-based technique [3], respectively.

In many early works, SLAM-based approaches have been developed for robots [4], autonomous cars/drones [5], AR/VR devices [6], etc. These approaches are expected to acquire a three-dimensional (3D) map of the working environments. On the one hand, range sensors are required for the acquisition of 3D maps for a UAV, such as laser-scanner, optical flow sensor, stereo camera and RGB-D camera. However, heavy sensors-based methods are infeasible, especially when it comes to flying platforms that are small in weight and power consumption. Furthermore, optical flow sensor and stereo camera are not suitable for long-range obstacle avoidance. On the other hand, the updating process of SLAM's 3D map is slow and does not fulfil the real-time requirement of fast manoeuvre. Moreover, there are critical issues like visual aliasing and dynamic scenes that can drive the system to unrecoverable errors.

Similar to SLAM-based methods, SfM-based methods prefer to calculate the depth of scenes and the path of cameras, which is a complex process. As such, the speed of quadrotor is often limited. Since perception and control are relatively independent parts, the possibility of positive feedback between them has been hindered, as discussed in [3].

Learning-based methods illustrate advanced performance in various machine vision tasks. Since the surroundings can be intuitively perceived through the images captured by UAV, machine vision-based method is suitable for robots to avoid obstacles [7, 8]. Researches on autonomous UAVs has employed deep learning algorithms [9, 10, 11], and shown notable positive results. Indeed, the development of machine learning enhanced the performance of the visual-based obstacle avoidance method, due to the learning-based perception approaches, which enabled feature extraction by tuning the parameters in training.

In this direction, several reinforcement learning (RL)-based approaches are developed to improve the robustness of obstacle avoidance. Sanket et al. [12] proposed a method to fly through unknown gaps with a monocular camera and onboard sensing for the presentation of experimental results. Kaufmann [13] proposed a deep-learning-based approach which showed fast adaptation to an approximated map. Singla et al. [14] designed a UAV control algorithm to combine information obtained over a period of time, which could improve the accuracy of the decision. However, this approach limits the scope of applications as the UAV needs to fly in the same workspace for several times, so it hinders the application of the UAV's operations in safety-critical environments.

In contrast, supervised-learning methods offer a more viable way to learn effective flying policies, but these methods still leave the issue of collecting enough expert trajectories for imitation. Collision-avoided trajectories by human expert pilots are of necessity to teach the robotic platform how to behave in dangerous situations [3]. Additionally, as pointed out by [3, 10], in order to ensure that the UAV has a better knowledge of how to control the direction in flight, the steering angle can be provided by expert pilots. While using deep learning in flying tasks, the importance of the computation complexity and tracking accuracy have to be considered. In particular, the obstacle avoidance task aims at obtaining the best accuracy under a limited computational budget, provided by certain hardware (e.g. a mobile device). Considering the requirements, the front-facing monocular camera can be used as the primary sensor for the UAV, which has a low cost in terms of computation and power.

Recently, as an effective alternative, deep learning offers a way to connect perception and control, which achieves impressive results [15, 16, 17, 18]. Giusti et al. [10] proposed a CNN-based algorithm, with inputs as one image from the front-facing camera and outputs as one of three commands, i.e. go straight, turn left and turn right, and its advantages for flying a quadrotor through forest trails was demonstrated. Three head-mounted (point straight, left and right) cameras are used to collect data , and the label of each image is the direction of the camera taking this image. This means that the directional control output is also learnt by the network and tailored to one particular environment [19].

In summary, to achieve the objective of obstacle avoidance without a 3D map, it is essential to control the yaw angle and the forward/backward speed of the UAV. For this purpose, an automatic obstacle avoidance system based on CNN is proposed for UAV in this paper. The CNN-based network works for predicting the quadrotor's steering angle and the collision probability as a front-end stage, and the control commands are obtained by mapping in a back-end stage. The processing speed is taken into account when evaluating the network that is to be used. The system is shown to perform well in both outdoor and indoor environments. The research contributions are highlighted as follows.

1. Several representative state-of-the-art networks are investigated, as well as a novel model achieving credible results and satisfying the real-time requirements of UAV. The effectiveness of the proposed model using the real-world driving datasets and collision datasets is demonstrated.
2. The first-order Butterworth low filter was adopted to map the predicted results from the networks to the control instructions, which makes the control process more smooth and sensitive.
3. A pitch angle control mechanism was introduced in a deterministic arbitration scheme, to enable the UAV to perceive the environment in a 3D space. In the method, another prediction processing with the top part of an image is mapped to pitch angle instead of controlling forwarding speed. Unlike most of other works where the system continuously estimates the pitch angle, our mechanism is triggered only when the UAV stops with the collision probability approximately equal to 1.

The obstacle avoidance system proposed in this work represents a good fit for the obstacle avoidance tasks. The CNN model and filter, which are used to generate control commands, are efficient in enabling real-time, smooth and reliable response to the UAV's situation. Attributing to the datasets from the various scenes, different in lighting, location, etc., the proposed system is versatile for the multifarious environments. Meanwhile, three degrees of freedom of UAV are controlled simultaneously to avoid any dead ends, and to make the flight more flexible.

2. Related Works

Obstacle avoidance is an important part of research on autonomous quadrotors, which mainly involves the use of GPS. The GPS's real-time data loss is prone to fall in this situation of indoor and outdoor environments. In comparison, due to the low cost of visual sensors, the vision-based obstacle avoidance technique is widely used in industries. This technique has been developed for years and has made dramatic progress, especially with the recent developments of computer vision to process the image information acquired by the camera without using GPS. Combined with the data acquired by other sensors of the UAV, the current position of the UAV can be calculated precisely with a relatively small computational load.

In this section, robotic learning and obstacle avoidance in dynamic environments that are related to our work are briefly described.

2.1. Deep Learning for Robots

Exploration of a drone's obstacle avoidance with deep learning is found to be effective, so the number of research works on developing learning schemes using raw sensory data and deep neural networks has significantly increased in the past few years. Among these works, studies based on supervised learning have achieved encouraging results [10].

A suitable network is essential to make accurate and fast decisions during the flight and to respond to the operating environments for a UAV. Hence, a lightweight model is of vital importance to achieving automatic obstacle avoidance of the UAV. In recent years, CNN for mobile devices has gained tremendous growth and many advanced efficient networks were proposed [20, 21, 22, 23, 24],

where the number of parameters and the computational burden of these models were remarkably reduced under the premise of ensuring a good performance. Squeezenet [23] proposed a fire module that only used 1/50 of the parameters, achieving the same correction rate as AlexNet [25] on ImageNet. Mobilenets [20] used depth-wise separable convolution, width multipliers and resolution multipliers to reduce the size and latency. Shufflenet [26] proposed channel shuffle and employing point-wise group convolution, whose complexity was less than Mobilenets [20]. Mobilenet-V2 [24] inverted residual block and ReLU6 achieved the same accuracy as ShuffleNet [26] with fewer parameters. Shufflenet-V2 [27] designed the network structure based on four guidelines, which was shown to be faster with an accuracy comparable to MobileNet-V2 [26]. However, the computational burden of these models is still high when applied to the UAV's obstacle avoidance. Also, it is proved that part of the computation is actually not required since the blocks with a smaller number of layers could yield reliable results for the UAV's obstacle avoidance.

2.2. Obstacle Avoidance in Dynamic Environments

A common approach for obstacle avoidance in dynamic environments is to use specific control strategies to avoid collisions for robots [28] by treating dynamic factors, such as pedestrians, as dynamic obstacles. Sophisticated models are required to build for dynamic obstacles and actions based on possible states of robots also need to be defined to avoid collisions. However, the real-world environment is extremely involved in the sense that dynamic obstacles can move in a variety of ways, so it is difficult to include all possible states of robots. As a robot encounters a situation that does not exist in the model, the robot may fail in finding a viable action, which may cause the system to lose control. Several studies [20, 29] developed various methods, which mainly focused on learning-based methods, to solve this problem. Unfortunately, a training process requires an explicit model for a dynamic obstacle, so different models are required for different factors that are difficult to predict beforehand. Hence, in the model to be established in this paper, dynamic factors will be included in the datasets that are in the form of images.

On the other side of the spectrum, various approaches have focused on the perception task, employing custom motion planning schemes to determine the robot's action based on the perception output. In [19], a depth map is predicted for each monocular image captured by the drone's onboard camera, using a CNN trained on RGB-D data. A deterministic arbitration scheme is employed to steer the UAV away from obstacles by controlling its angle on two rotational degrees of freedom (DOF) based on the generated depth map. However, thousands of image-depth map tuples collected from a depth sensor are used to train a CNN to determine the depth from a single image. This method relies on depth cameras, which are not always available.

In this paper, to detect a trajectory of avoidance in the UAV's environment, we train a CNN to run inference on the input image, a classification branch and a regression branch, sharing the same feature extractor with the regression stream and trained on datasets with dynamic factors. Autonomous driving datasets' steering angles are used as labels while training the regression model, and the classification branch's datasets are labelled as positive or negative based on the distances to the surrounding vehicles/objects. In contrast to datasets labelled

with depth images, these improve resilience to environments, thereby allowing simple and easy collection of indoor/outdoor scenes for use in training.

3. Intelligent Obstacle Avoidance Method Based on Convolutional Neural Network

For autonomous UAV obstacle avoidance, some basic requirements for image processing include the following characteristics.

1. Accuracy. Specifically, the prediction should achieve nearly 100% recalls with high accuracy.
2. Speed. The prediction should ensure real-time processing and a fast inference speed to reduce the latency of the UAV control loop.

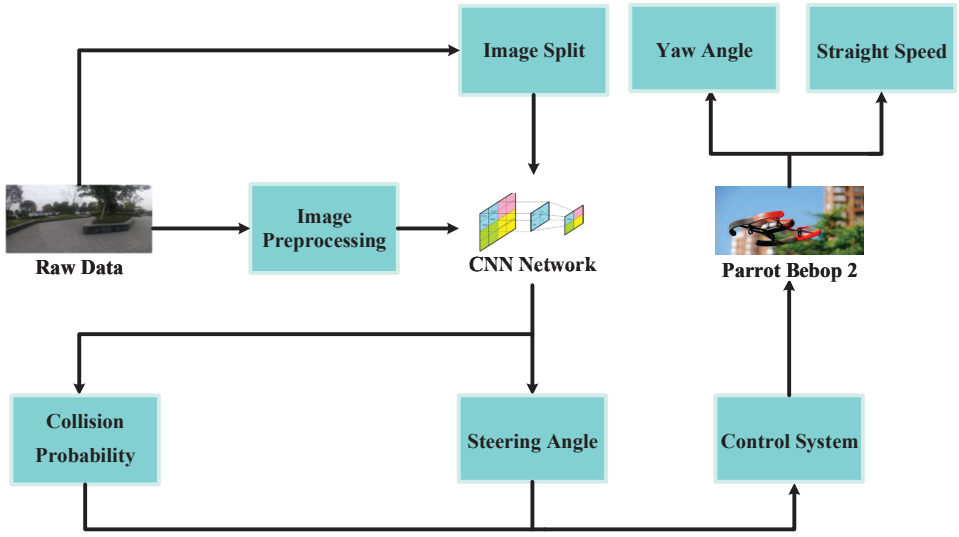


Figure 2: **Architecture of UAV's automatic obstacle avoidance system.** Initially, image is obtained from the front camera and processed to increase the quality. Collision probability and steering angle are predicted by CNN Network with the preprocessed image. When the prediction of the collision probability is nearly to 1 and steering angle is small, the image would be split into top and bottom parts, and whether the drone can leap the obstacle by raising the altitude is determined by the network's prediction with the top part of image. The stability of bebop's flight is being controlled by a control system that maps the predicted results to the instructions.

Meeting the above requirements, obstacle avoidance is achieved by efficient and light-weight networks in this paper. To achieve it in a highly dynamic scenario, e.g., in a pedestrian-rich environment, the operating state of perception is split into two different categories: $\mathcal{G} = \{collision\ probability,\ steering\ angle\}$. Each category has a specific policy that maps a state of UAV control commands, which consist of linear velocity $\dot{x}$, angular velocity $\dot{z}$ (yaw) and angular velocity $\dot{y}$ (pitch) [30]. The architecture of the obstacle avoidance system proposed in this paper consists of two parts: hardware integration, prediction and control software, as shown in Figure 2.

Moreover, the UAV's model and coordinate system are shown in Figure 3. The flight states are observed by the UAV's front-facing camera. To train a policy of CNN model, the datasets labelled by collision probability are ulitized. Despite the separate policies, two operating states share the same network structure. Therefore the added computation is minor. Each step will be introduced in details in the remainder of this section.

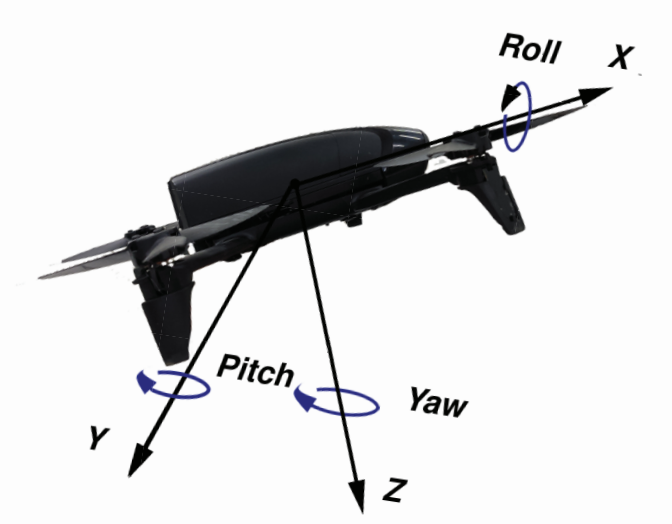


Figure 3: **UAV and coordinate system**

3.1. Framework Overview

A block diagram of our training system is shown in Figure 4. Images are fed into the proposed model which computes a steering angle and a collision probability. In the prediction part, the result found by our network is compared with the desired one. Weights of the model are adjusted to make the prediction outputs close to the desired steering angle and straight speed commands. The weighting adjustment is accomplished using backpropagation.

3.2. Input Image for CNN Model

The CNN model's accuracy and processing speed are influenced by the size of data. Due to the lack of knowledge about the environmental data that affect the computation and memory costs, the image transferred to the network is compressed to $(200, 200)$. A median filter and a Gaussian filter are utilized to filter out noises. It is noted that the upper part of an image, which is captured by the front-facing monocular camera, has less focus and less environmental features with respect to the lower and middle parts. In this paper, one-eighth of the image from above is cropped in the image compression process according to the observation during data collection.

3.3. Learning Approach

3.3.1. Convolutional neural networks

CNN is a neural network that can be used to extract visual features from images. Compared with traditional methods using hand-crafted features and

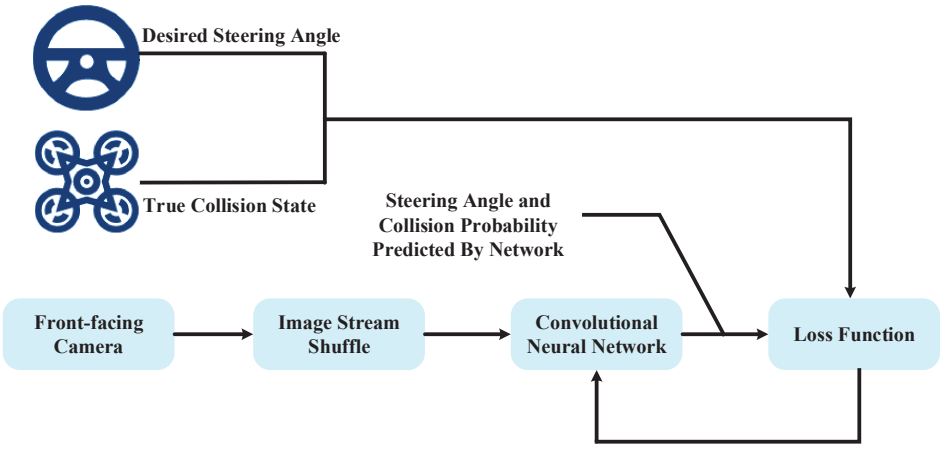


Figure 4: **Neural network training**

fully-connected networks, CNN-based methods have achieved significant advancement in the field of computer vision. Neurons in CNN are locally connected in spatial dimensions, and all are connected in channel dimensions [31].

CNN is composed of a series of nonlinear layers. Considering a single image x_0 passing through a convolutional network of L layers, each layer implements a nonlinear transformation H_l. The nonlinear transformation is commonly defined as a convolution followed by a rectified linear unit (ReLU), batch normalization (BN) [32] or dropout [33].

For each convolutional layer l, a set of filters are learned to express local spatial connectivity patterns along input channels. In other words, convolutional filters are expected to be informative combinations by fusing spatial and channel-wise information together within local receptive fields. By stacking a series of convolutional layers interleaved with non-linearities and downsampling, CNNs are capable of capturing hierarchical patterns with global receptive fields [31].

The tensor of output by the CNN is actually a distributed and sparse third-ordered expression of the input image. Distributed expression theory indicates that the concept formed by neural networks is based on the algorithm of each neuron involved, which means that CNN is a multi-concepts multi-neurons model [31].

3.3.2. Network Architecture

Because of the critical real-time requirement of the UAV control, the time period between sending the flying status and receiving the control command is subjected to a limit. In general, the shorter the period is, the better it is. Thus, as discussed in Section 2.1, the network used in this work is required to satisfy the light-weight requirement.

To evaluate the computation complexity of a model with image-type datasets, frames-per-second (FPS) and the number of float-point-operations (FLOPs) are widely used. Several important factors that have considerable influences on speed are not taken into account by FLOPs [27]. Also, FLOPs and FPS could be described with different values in various software and hardware settings. To

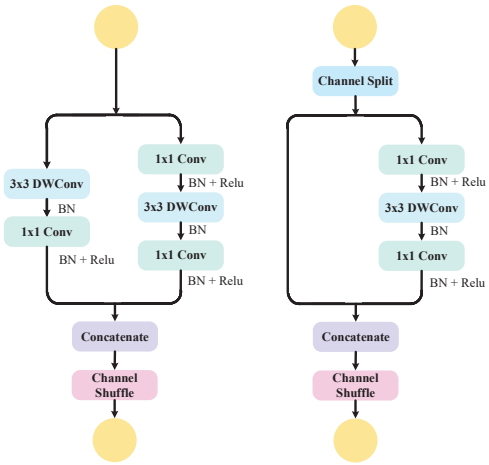


Figure 5: **Building blocks**

solve this problem, the number of parameters and memory access cost (MAC) are used to evaluate the computation complexity of the model. MAC is vital to evaluate performance, as discussed in [34]. According to [27], to achieve the practical design of an efficient model with a relatively lower MAC, the following guidelines are suggested as following:

- equal channel width minimizes MAC;
- excessive group convolution increases MAC;
- network fragmentation reduces the degree of parallelism;
- element-wise operations (such as adding activation function, adding bias, etc.) have non-negligible effects to MAC.

In this paper, the model is initialized by a 3×3-stride-2-convolution-layer with 2-stride max pooling. The building blocks have three storeys with different sizes: the size of the first storey is 28×28, the second is 14×14, and the last is 7×7. The blocks are adopted from Shufflenet V2 and are shown in Figure 5. At the beginning of each block, the spatial downsampling unit is used to change the size of output, and it connects basic units. Considering real-time and accuracy requirements, the basic units used in each block are defined as $2, 4, 2$. After the last channel-shuffle layer, the architecture splits into two different convolution layers, namely, a global-max pooling layer and a fully-connected layer.

As an adaptive learning rate optimization algorithm, Adam is chosen as an optimizer to train the network with a batch size of 64, the learning rate of 0.001, and the learning rate decay is set to 0. The models are trained for 130 epochs. To avoid overfitting to the training data, Batch normalization, regularization, and Dropout are introduced into the network. Specifically, a group ridge regularization of 10^{-5} is added in each convolution layer, the convolution layers in the building blocks are all followed by Batch normalization layers, also, a Dropout operation is introduced in the front of the fully connection layer. To further reduce the overfitting, the training process may be stopped when the network show insignificantly improved in accuracy in five epochs, this method is similar

to early stopping. The accuracy on the training dataset of 97.9% compared to 96.6% on the testing dataset, shows that the network has a generalization on the dataset.

While one of the branches represents yaw prediction, the other one makes a strategic decision based on the UAV's location relative to collisions. In specific, collision prediction and yaw angle prediction have two different fully-connected layers. The structure of the CNN model is shown in Figure 6.

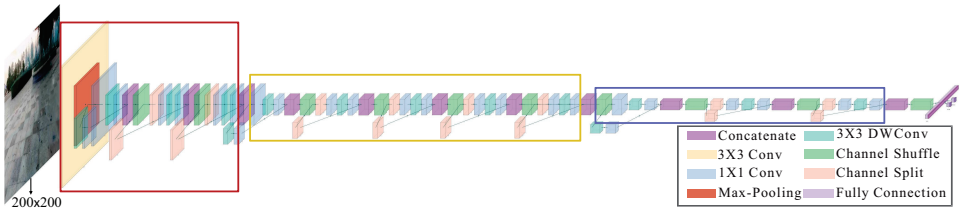


Figure 6: **Architecture of the CNN model.** The input image is preprocessed and resized into 200×200 pixels. Except the concatation and the channel-shuffle layers, the layers in red, yellow, blue boxs have $24, 48$ and 96 channels, respectively.

A regressive model based on mean-squared error (MSE) is used for the prediction of yaw angle.

$$MSE(X, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2} \tag{1}$$

where X denotes all the feature values of datasets, h is the prediction function of CNN system, i.e., the CNN system outputs a predicted value $h(x^{(i)})$ when the system is given an instance's feature vector $x^{(i)}$, m is the number of images in the datasets, $x^{(i)}$ and $y^{(i)}$ denote respectively the feature values (In this paper, it represents the preprocessed image itself) and label of the i-th image. Binary cross-entropy (BCE) loss function is used to train the collision prediction. In this task a two-class problem is described as follows.

$$L_{Logistic(p,y_i)} = -y_i \log(p(y_i)) + (1 - y_i) \log(1 - p(y_i)) \tag{2}$$

where y_i is the label (1 for collision and 0 for no collision), and p is the predicted probability of the collision images out of all images in the datasets.

Since the model (2) is single-input and multiple-output, the solver is required to find an optimal solution for the two tasks at the same time. The gradients magnitude of the regression is different from it in the classification task. Naive joint optimization poses serious convergence problems. Indeed, the gradients in the two loss functions (MSE for regression task, BCE for classification task) without adjusting the weights has a negative effect on the convergence of the training process, which will lead to the adverse consequences[35]. Furthermore, it is confirmed that supplying the tasks in a meaningful order may lead to improved performance and better convergence[36]. Therefore, it is necessary to impose weights for two loss functions when training the model. Considering that the steering angle needs more epochs to be optimized, the weights used in

[3] are adopted by defining

$$L_{tot} = L_{MSE} + \max(0, 1 - \exp^{-\frac{e_i - e_0}{e_0}}) L_{BCE} \tag{3}$$

where L_{tot} is the total loss value for training, e_i is the current epoch, e_0 is a value set as 30, and L_{MSE} and L_{BCE} represent loss values corresponding to the regression and classification tasks, respectively.

3.4. Deterministic arbitration

The two-values output of the network can be mapped as the UAV's control commands to make the UAV automatically avoid obstacles with forward velocity v_t, yaw angle θ_t and pitch angle ϕ_t. Specifically, v_t is obtained by mapping the probability of collision p_t, which is provided by the output of the network. Since the probability of collision is in the range of $[0, 1]$, the UAV is expected to fly with speed between the maximal speed V_{max} and 0.

To ensure the stability of the flight, the control command must be smooth. Moreover, sensitive obstacle detection is essential for UAV automatic flight system. For these purposes, three classic filters, i.e. first-order filter, second-order filter and first-order Butterworth filter, have been tested. The first-order Butterworth filter performs more sensitive detection when the state changes, and it provides smooth control commands, for which the results will be reported in the experiment. Butterworth filter, also known as the Wagner filter, is characterized by its flat passband attenuation features. Also, it is widely used due to its simple design and superior performance. The first-order Butterworth low-pass filter [37] is given in terms of the transfer function $|H(jw)|^2$ as:

$$|H(jw)|^2 = \frac{1}{1 + (\frac{w}{w_c})^2} \tag{4}$$

where w_c is the cut-off frequency. This filter has the following properties:

- $|H(jw)|^2 = 1$ is the maximum, when $w = 0$ and the curve has maximum flatness;
- $|H(jw)|^2$ is a monotonically decreasing function of w and there are no fluctuations in amplitude.

The scaled steering s_k, the output of the yaw angle prediction part, is mapped to the rotational motion of the UAV around the vertical axis of the body coordinate system, i.e. yaw angle θ_k. In particular, s_k in the range of $[-1, 1]$ is required to convert into a desired yaw angle θ_k in the range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$. As the control of steering angle does not need to be sensitive to the environment changes like the forward speed control, a low-pass filter is used for simplicity. As such, the smooth, continuous control command is defined in the following:

$$\theta_k = (1 - \beta)\theta_{k-1} + \beta \frac{\pi}{2} s_k, \quad 0 < \beta < 1. \tag{5}$$

It should be mentioned that obstacle avoidance should be performed, when the predicted collision probability of the image at the current time Img_t exceeds a certain threshold (set as 0.9). Img_t is divided into top and bottom parts, which

have the same size. A warping with context padding ($p = 16$ pixels) transforms the top part into valid CNN model inputs. A 30 degree pitch angle command will be transmitted to the UAV when the collision probability of top parts is lower than a certain threshold (set as 0.3).

3.5. Data Collection

(a) Night images (b) Images collected during the day

Figure 7: **Samples of collected images** Images are extracted into two categories: images depicting the enviroments where UAV may function normally (green box), and images showing the UAV may collide with the environments (red box).

The datasets are divided into two distinct categories, one labelled with yaw angle and the other involving involves the collision information.

The UAV's yaw angle can be resembled to the self-driving-car's steering angle by projecting the quadrocopter z-axis into the horizontal plane. Two large-scale image autonomous driving datasets, containing over 1,200,000 frames collected from Comma.ai [38] and Udacity [39], are used to train the yaw angle prediction. These datasets acquired contain video clips captured by a single front-view camera, located similarly to the UAV front-facing camera and mounted on the windshield of the vehicle. Alongside the video data, a set of time-stamped sensor measurement is contained, such as the vehicle velocity, acceleration, steering angle, GPS location and gyroscope angles. In this work, the regression branch is trained using the image of front-facing camera and the corresponding steering angle only, on account of the similarity of the angle between the camera mentioned before and the monocular camera of UAV. Note that the estimates of the interpolated measurements of the sensor logs are used to deal with the problem that the sensor logs are out of sync with the time-stamps of video.

The datasets of collisions initially used to train the network are 45,000 images from the RPG collision data [3], which are collected in urban environments by cyclists, labelled as positive or negative based on the distance between the camera to its nearest objects. The dataset is then enlarged with 20,000 further samples from the experiments based on our networks trained by datasets mentioned above. Positive data need to be collected manually on account of nonequilibrium between positive and negative sample attributed to a manually stop of UAV when it is about to collide. Figure 7 depicts some samples of collected images. Datasets used in this paper are summarized in Table 1.

Table 1: **Datasets used in two prediction models**

	Datasets	Frame	FPS	Hours	Condition	Location	Lighting
Yaw angle prediciton datasets	Comma.ai	522,434	$20Hz$	$7hrs$	Highway/Urban	CA,USA	Day/Night
	Udacity	80,180	$20Hz/30Hz$	$8hrs$	Urban	CA,USA	Day
Collision prediction datasets	rpg-collision	32,000	-	-	Urban	SE	Day
	SW-collsion	49,350	$20Hz$	$6hrs$	Urban	China	Day/Night

Table 2: **Different forms of images with our model**

		Collation prediction		Steering angle prediction	
Images' scale	processing speed	Accuracy	F1 score	RMSE	EVA
RGB	$0.0212s$	0.9906	0.9367	0.1057	0.7563
Grayscale	$0.0053s$	0.9943	0.9610	0.1009	0.7709

Table 3: **Image classification and regression results**

Model	EVA	RMSE	MAE	R2	Avg. accuracy	F1-score
mobilenet v2	0.447	0.223	0.134	0.244	0.928	0.831
Resnet50	0.985	0.037	0.023	0.986	0.973	0.913
Xception	0.981	0.032	0.022	0.985	0.945	0.868
Our network	0.974	0.050	0.036	0.972	0.966	0.907
Dronet	0.901	0.095	0.066	0.876	0.935	0.861

Model	Num. parameters	FLOPs	Processing time (fps)	
			GPU	CPU
mobilenet v2	0.30M	61M	183	96
Resnet50	23.5M	3.8G	133	27
Xception	0.79M	242.6M	143	119
Our network	0.30M	25.9M	157	129
Dronet	0.32M	35.5M	148	122

4. Experiments

To demonstrate the accuracy and efficiency of the proposed framework, we test it on self-driving vehicle datasets. The grey images and RGB images are used to train the models, respectively. The average run-time for our model took slightly more than a day to train datasets. The model features are compared with the state-of-the-art architectures, including Xception and ResNet. The whole system is tested in real environments. In this section, the platform for the experiment is introduced and the run-time of the system is analysed. Quantitative experiments with comparisons to the states-of-the-art show the advantages of our system.

4.1. Evaluation of Regression and Classification Tasks

To quantify the performance on steering prediction, 4 indices, i.e., mean-absolute error (MAE), root-mean-square error (RMSE), explained variance score

(EVS) and coefficient of determination (R^2), are used. They can be calculated respectively as follows:

$$MAE(X,h) = \frac{\sum_{i=1}^{m}(|h(x_i) - x_i|)}{m}, \tag{6}$$

$$RMSE(X,h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2}, \tag{7}$$

$$EVS(h,Y) = 1 - \frac{Var\{y - h(x)\}}{Var\{y\}}, \tag{8}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{9}$$

where SS_{res} is the difference between predictions and average value of predictions, and SS_{tot} is the difference between labels and average value of labels.

Accuracy, receiver operating characteristic (ROC) curve and F1-score are used to evaluate the quality of the model, where F1-score is defined as:

$$F1 = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{10}$$

where TP, FN, FP and their relationship with ROC are described in Figure 8.

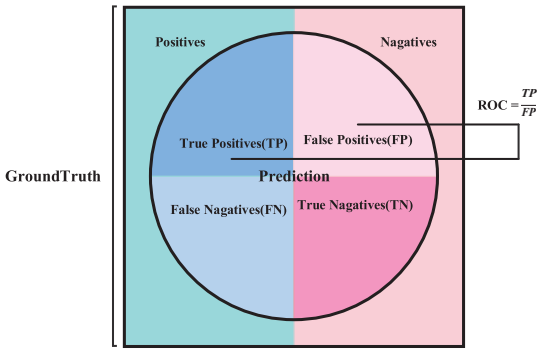


Figure 8: **Confusion matrix**

4.2. Comparison with Other Networks

Several architectures of networks have been tested to choose the most suitable one for autonomous navigation of the UAV. Performance of our network and two representative state-of-the-art networks are analyzed in this work. Besides the evaluation methods mentioned in Section 4.1, the running speed and the number of parameters involved in each network are also reported in Table 3. While the same settings are used for different models, the results show that our network outperforms the others with a significant improvement. In particular, our network surpasses Dronet with less FLOPs and achieves a better balance between the performance of two branches compared with Xception.

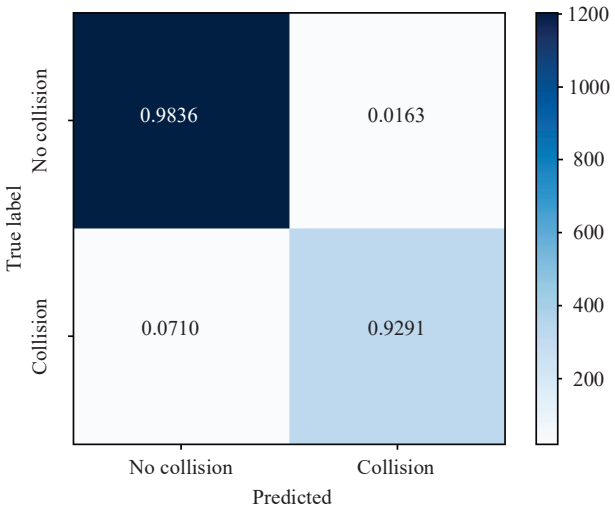


Figure 9: **Confusion matrix of collision**

4.3. Comparison with Different Forms of Images

As explained in Table 2, the processing speed, accuracy and F1-score for collision prediction of an image in different forms are compared. Meanwhile, the RMSE and EVA for steering angle prediction between datasets in grayscale and datasets in RGB are also compared.

4.4. Visualization of high-dimensional data model inspection

The performance of the proposed technique has been demonstrated by the quantitative evaluation. Furthermore, the trained model is tested. The high-performance CNN may be interpreted by visualizing the inner situation of networks, through the work by Selvaraju et al. [40]. In this work, this method is used for examining the performance of the model as well. The weights of the last convolution layer are utilized to support Grad-CAM to locate the parts of interest. The heatmap illustrating the locations of the sensitive part in the image is shown in Figure 10. The color depicts the sensitivity of that part in the images. It is obvious that the model is sensitive to surrounding vehicles/objects that may be collided. One inevitable situation is that the location of the sensitive area has a small excursion with the objects. The appearance suggestes that the regression branch influences the behavior of networks since the network is shared by two branches.

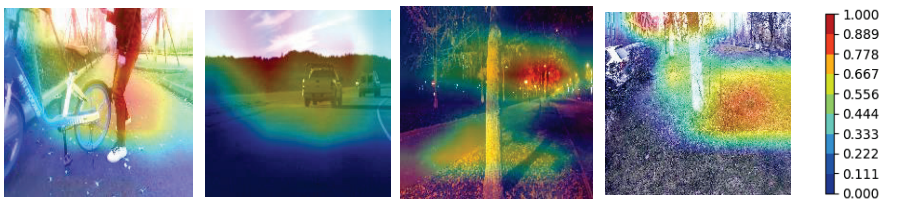


Figure 10: **Heatmaps of collision prediction.** Images from left to right show the environments of dynamic obstacles, a city road, a tree at night and a jungle.

4.5. Control commands processing

In this subsection, the deterministic arbitration methods mentioned in 3.4 are compared via 564 pairs of data taken from the collision datasets. To make the results more convincing, the datasets contain two types of state changes, i.e. from no-collision to collision and from collision to no-collision. From Figure 11, it is found that the first-order Butterworth filtering has the highest response speed when the state change occurs and is able to provide smooth control commands when applied to the original data.

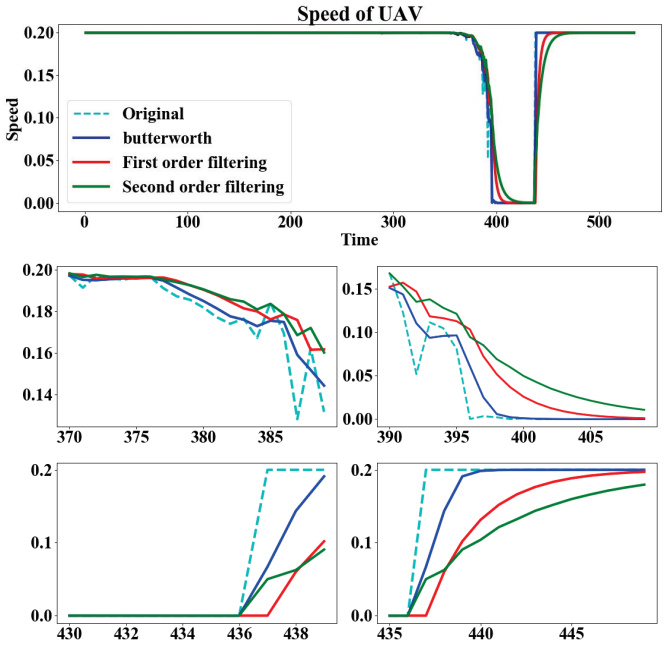


Figure 11: **Smoothing and truncation of control commands using various filters.** The bottom figures show more details of the top figure at different states: close to obstacles (top left), keeping a safe distance from obstacles (bottom left), leaving the obstacles and returning to normal flight away from obstacles (bottom right).

4.6. Experimental setup

The proposed framework is evaluated on a modified UAV, Parrot Bebop 2, and the effectiveness of this low-cost system for complex tasks of flying in outdoor/indoor scenes in urban environments is verified. Using Wifi, Bebop can communicate with the ground control station with the ROS package and Bebop-Autonomy that sends the control command to the UAV, and obtain the flight information of the drone, such as images from the front camera. The UAV's onboard monocular camera's images are collected with a 640×368 resolution at 30 fps. To guarantee images veracity, the camera is calibrated by initial images with the cameracalibrator node of ROS. All the experiments are conducted on a laptop running Ubuntu 14.04 with 8GB RAM, an Intel Core i7 2.6GHZ CPU and a NVIDIA 1080 GPU. The same equipment is used to build the ground control station.

4.7. Runtime analysis

To analyze the effects of the system, the time-consumption and varying delays, including image and control transmission delays of three major components (Image preprocessing, CNN processing and control mechanism) are considered. The correlational analysis of time-consumption is presented in Figure 12. According to the measurement of time consumption, there is no significant difference whether to add the cost of image preprocessing (970fps) and control mechanism parts. The consumption time of the proposed framework, as can be seen from the table, is acceptable for automatic obstacle avoidance of UAV.

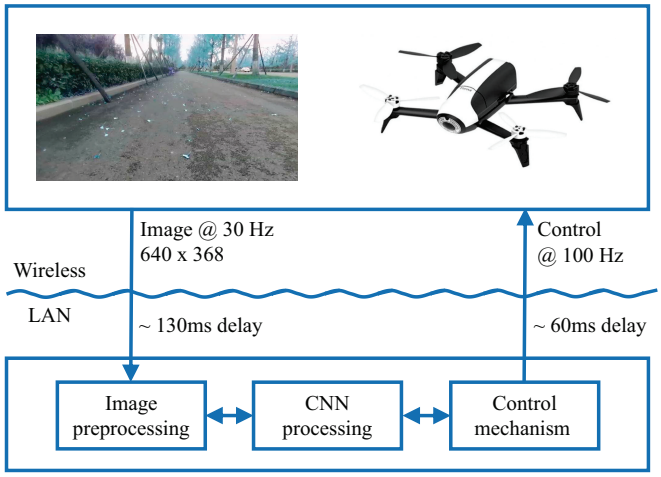
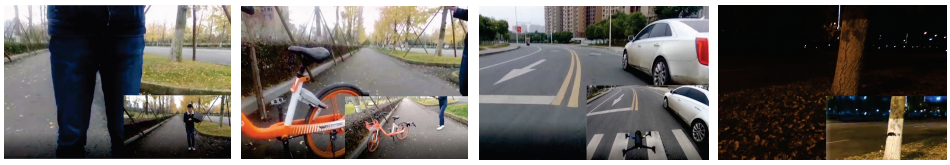


Figure 12: **System run time analysis**

4.8. Performance of the model in controlling UAV in real indoor/outdoor environments

The most intuitive way to assess the achievements of system is measuring the performance of flight in highly dynamic scenarios, such as passing vehicles, bicycles and pedestrians, as shown in Figure 13.



(a) (b) (c) (d)

Figure 13: **Obstacle avoidance is achieved by the UAV.** In each subfigure, the image in the lower right corner shows the environment of the UAV, while the image on the left is taken by the front-facing camera of the UAV. (a) Avoidance of a pedestrian; (b)Avoidance of a bicycle; (c)Avoidance of a car; (d)Avoidance of a tree at night.

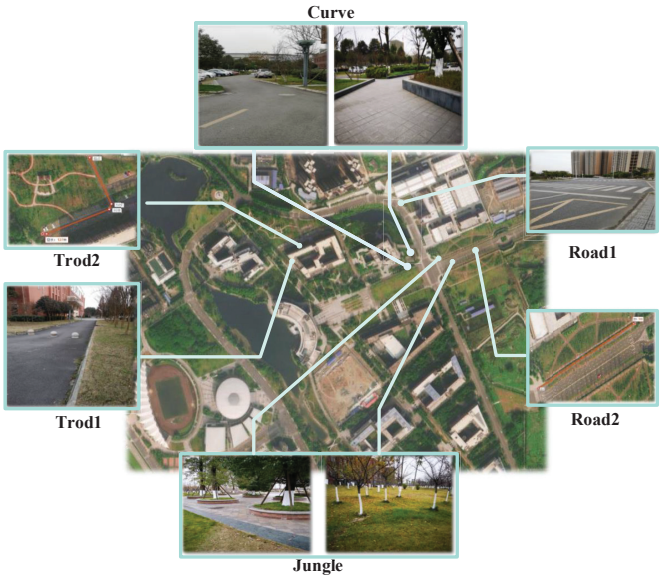


Figure 14: **A map of the campus, with marked areas indicating testing environments.** These areas are good representations of the urban environment. Pictures around the map present some featured samples of these areas.

4.8.1. Outdoor

To evaluate the generalization performance, eight scenes including day, night, outdoor and indoor that are not included in the training datasets have been chosen as outdoor experimental environments. Figure 14 illustrates the environments of tests. For each sub-task, the performance of the system with grayscale images is compared with the system with RGB images and Dronet.

Table 4: **Performance of the model in controlling the UAV in real environments**(Unit:m)

	Road1	Road2(day)	Trod1	Trod2	Curve	Jungle	Road2(night)
model with grayscale	368	154	75	127	32	90	45
Dronet (Resnet8) with grayscale	319	154	75	79	32	48	19
model with RGB	342	142	63	104	27	54	-

Road test. With self-driving datasets used for training of the model, the UAV has a reliable performance when tested on the scene Road1 and Road2. Three experiments are carried out in each environment, and similar good experimental results are obtained as summarized in Table 4. The average flight distances of system with grayscale with 368m on Road1, and 154m or more on Road2, indicate that the proposed automatic obstacle avoidance system has a robust fitting ability for highway scenes and it can be easily readapted to a new road that includes a straight line or a bend.

Complex trod test. The proposed system is tested in two complicated trod environments. There is no marked lane in such environments, which may affect the decision of the network, so it is more challenging for navigation. Results

confirm that the UAV can fly autonomously in a distance, although the lawns aside the trod have an inevitable impact on the obstacle avoidance system. Our model with grayscale images achieved the best performance, as summarized in Table 4.

Continuous curve test. In this environment, the challenge is manoeuvre through continuous and irregular curves. Experimental results show that compared with the other two methods, our system with grayscale images can smoothly pass through a continuous fold line with results summarized in Table 4.

Jungle flight test. For further testing, the generalization capabilities of the system for complex scenarios are tested by flying the UAV in a jungle. For this scenario, as the model does not have the same datasets for training, the UAV using the other networks is unable to navigate to a target, but our network with grayscale images found a relatively better route, with experimental results recorded in Table 4.

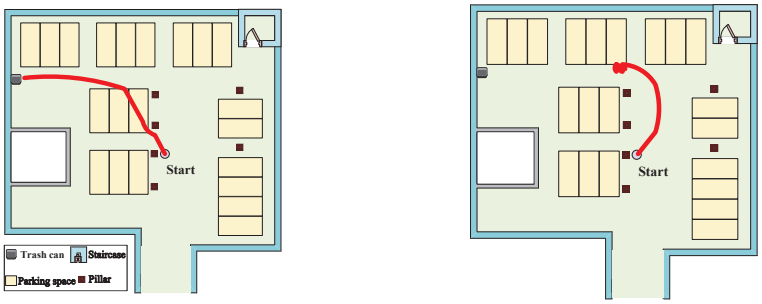
Night test. The experiment at night is designed to verify the system's capability to deal with different lighting conditions. As shown in Table 4, the system performs 45m flight after training with datasets in different lighting conditions and has a considerable increase in capability against other systems. Furthermore, this is an advantage compared with the initial system, which achieved a flight distance of nearly 25m. The better result may attribute to more training samples, which were collected from the experiments on the initial system (experimental data of Road2 is excluded). The experiment shows that the performance of the proposed system with grayscale images is comparable to the other two mechanisms in outdoor environments. Indeed, in every environment test, our system can respond to situational changes at any moment. A key observation is the failure of collision avoidance to bushes based on the other methods, as bushes give the model wrong information that the surrounding objects might be very close to UAV. However, our method can fly over the bushes due to re-prediction on the top part of the image.

4.8.2. Indoor

The system with different forms of images is tested in an underground garage, to evaluate the performance in indoor environments, shown in Figure 15. The single most striking observation from the comparison was the experiment relying on grayscale images performs slightly better than that on RGB images. This discrepancy could be attributed to the delay caused by time consumption, i.e., processing an RGB image takes longer than processing the grayscale image in transmission, image preprocessing, and network operation. According to the comparison, we can infer that the RGB image is not necessary for automatic obstacle avoidance, although it contains more environmental information than grayscale images.

5. Conclusion

This paper presents an automatic obstacle avoidance system for a UAV to fly safely and efficiently in indoor/outdoor environments. The datasets used in the experiments include datasets from two large-scale real driving scenarios and



(a) Route with grayscale images (b) Route with RGB

Figure 15: **An underground garage environment, in which the red curves represent paths with different forms of images.**

UAV collision datasets collected by flying the UAV manually. The first-order Butterworth filtering is used to map the outputs of networks to UAV commands. Extensive experiments using five outdoor/indoor scenarios prove that the system proposed in this paper is efficient for UAV obstacle avoidance tasks due to its low sensor requirement, light-weight network structure, strong learning ability and environmental portability. The platform used for experiments relies on WiFi, which imposes restrictions on communication distance between UAV and ground control station. Therefore, an image/data transmitter of UAV based on 4G/5G mobile communication technology is being developed by our team, and a platform with the transmitter will be investigated in our future work.

References

[1] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, Orb-slam: A versatile and accurate monocular slam system, IEEE Transactions on Robotics 31 (5) (2018) 1147–1163.

[2] H. Alvarez, L. M. Paz, J. Sturm, D. Cremers, Collision avoidance for quadrotors with a monocular camera, in: Experimental Robotics, 2016, pp. 195–209.

[3] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, D. Scaramuzza, Dronet: Learning to fly by driving, IEEE Robotics and Automation Letters 3 (2) (2018) 1088–1095.

[4] P. Ardón, K. Kushibar, S. Peng, A hybrid slam and object recognition system for pepper robot, arXiv preprint arXiv:1903.00675.

[5] O. Mendez Maldonado, Collaborative strategies for autonomous localisation, 3d reconstruction and pathplanning., Ph.D. thesis, University of Surrey (2018).

[6] L. Chen, K. Francis, W. Tang, Semantic augmented reality environment with material-aware physical interactions, in: 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct), IEEE, 2017, pp. 135–136.

[7] T. W. Ubbens, D. C. Schuurman, Vision-based obstacle detection using a support vector machine, in: 2009 Canadian Conference on Electrical and Computer Engineering, 2009, pp. 459–462. doi:10.1109/CCECE.2009.5090176.

[8] F. Espinosa, M. R. Jiménez, L. R. Cárdenas, J. C. Aponte, Dynamic obstacle avoidance of a mobile robot through the use of machine vision algorithms, in: Symposium of Signals, Images and Artificial Vision - 2013: STSIVA - 2013, 2013, pp. 1–5. doi:10.1109/STSIVA.2013.6644903.

[9] N. Dinh Van, G. Kim, Fuzzy logic and deep steering control based recommendation system for self-driving car, in: 2018 18th International Conference on Control, Automation and Systems (ICCAS), 2018, pp. 1107–1110.

[10] A. Giusti, J. Guzzi, D. C. Cireşan, F. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, L. M. Gambardella, A machine learning approach to visual perception of forest trails for mobile robots, IEEE Robotics and Automation Letters 1 (2) (2016) 661–667. doi:10.1109/LRA.2015.2509024.

[11] X. Yang, H. Luo, Y. Wu, Y. Gao, C. Liao, K.-T. Cheng, Reactive obstacle avoidance of monocular quadrotors with online adapted depth prediction network, Neurocomputing 325 (2019) 142–158.

[12] N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermüller, Y. Aloimonos, Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight, CoRR abs/1802.05330. arXiv:1802.05330.
URL `http://arxiv.org/abs/1802.05330`

[13] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, D. Scaramuzza, Beauty and the beast: Optimal methods meet learning for drone racing, CoRR abs/1810.06224.

[14] A. Singla, S. Padakandla, S. Bhatnagar, Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge, CoRR abs/1811.03307.

[15] Y. Cheng, W. Zhang, Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels, Neurocomputing 272 (2018) 63–73.

[16] N. Rezzoug, P. Gorce, A reinforcement learning based neural network architecture for obstacle avoidance in multi-fingered grasp synthesis, Neurocomputing 72 (4-6) (2009) 1229–1241.

[17] B. Daachi, T. Madani, A. Benallegue, Adaptive neural controller for redundant robot manipulators and collision avoidance with mobile obstacles, Neurocomputing 79 (2012) 50–60.

[18] M. Turan, Y. Almalioglu, H. Araujo, E. Konukoglu, M. Sitti, Deep endovo: A recurrent convolutional neural network (RCNN) based visual odometry approach for endoscopic capsule robots, Neurocomputing 275 (2018) 1861–1870.

[19] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, L. V. Eycken, CNN-based single image obstacle avoidance on a quadrotor, in: IEEE International Conference on Robotics & Automation, IEEE, 2017, pp. 6369–6374.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, CoRR abs/1704.04861.

[21] F. Chollet, Xception: Deep learning with depthwise separable convolutions, CoRR abs/1610.02357. arXiv:1610.02357.
URL http://arxiv.org/abs/1610.02357

[22] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, CoRR abs/1611.05431.

[23] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size, CoRR abs/1602.07360. arXiv:1602.07360.
URL http://arxiv.org/abs/1602.07360

[24] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, L. Chen, Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation, CoRR abs/1801.04381. arXiv:1801.04381.
URL http://arxiv.org/abs/1801.04381

[25] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90.

[26] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, CoRR abs/1707.01083. arXiv:1707.01083.
URL http://arxiv.org/abs/1707.01083

[27] N. Ma, X. Zhang, H. Zheng, J. Sun, Shufflenet V2: practical guidelines for efficient CNN architecture design, CoRR abs/1807.11164. arXiv:1807.11164.
URL http://arxiv.org/abs/1807.11164

[28] J. Bi, T. Xiao, Q. Sun, C. Xu, Navigation by imitation in a pedestrian-rich environment, CoRR abs/1811.00506. arXiv:1811.00506.
URL http://arxiv.org/abs/1811.00506

[29] Y. F. Chen, M. Everett, M. Liu, J. P. How, Socially aware motion planning with deep reinforcement learning, CoRR abs/1703.08862. arXiv:1703.08862.
URL http://arxiv.org/abs/1703.08862

[30] N. K. Sinha, N. Ananthkrishnan, Elementary flight dynamics with an introduction to bifurcation and continuation methods, CRC Press, 2016.

[31] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[32] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, CoRR abs/1502.03167. arXiv:1502.03167.
URL http://arxiv.org/abs/1502.03167

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (1) (2014) 1929–1958.

[34] W. Wang, Z. Pan, Dsnet for real-time driving scene semantic segmentation, CoRR abs/1812.07049. arXiv:1812.07049.
URL http://arxiv.org/abs/1812.07049

[35] Z. Ren, D. Dong, H. Li, C. Chen, Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning, IEEE transactions on neural networks and learning systems 29 (6) (2018) 2216–2226.

[36] S. Ruder, An overview of gradient descent optimization algorithms, CoRR abs/1609.04747. arXiv:1609.04747.
URL http://arxiv.org/abs/1609.04747

[37] G. Bianchi, R. Sorrentino, Electronic filter simulation & design, McGraw Hill Professional, 2007.

[38] Comma.ai, Public driving dataset, https://github.com/commaai/research, accessed March 7, 2017.

[39] Udacity, Public driving dataset, https://www.udacity.com/self-driving-car, accessed March 7, 2017.

[40] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, D. Batra, Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization, CoRR abs/1610.02391. arXiv:1610.02391.
URL http://arxiv.org/abs/1610.02391